Hindawi
Complexity
Volume 2019, Article ID 3951495, 13 pages
https://doi.org/10.1155/2019/3951495

Research Article

Multilayer Architecture Model for Mobile Cloud Computing Paradigm

Higinio Mora,[1] **Francisco J. Mora Gimeno**,[1] **María Teresa Signes-Pont**,[1] **and Bruno Volckaert**[2]

[1]*Department of Computer Science Technology and Computation, University of Alicante, 03690 Alicante, Spain*
[2]*Internet Technology and Data Science Lab (IDLab), imec, Department of Information Technology, Ghent University, Belgium*

Correspondence should be addressed to Higinio Mora; hmora@ua.es

Received 9 November 2018; Accepted 27 January 2019; Published 11 February 2019

Guest Editor: Andrew Schumann

Mobile Cloud Computing is one of today's more disruptive paradigms of computation due to its effects on the performance of mobile computing and the development of Internet of Things. It is able to enhance the capabilities of devices by outsourcing the workload to external computing platforms deployed along the network, such as cloud servers, cloudlets, or other edge platforms. The research described in this work presents a computational model of a multilayer architecture for increasing the performance of devices using the Mobile Cloud Computing paradigm. The main novelty of this work lies in defining a comprehensive model where all the available computing platforms along the network layers are involved to perform the outsourcing of the application workload. This proposal provides a generalization of the Mobile Cloud Computing paradigm which allows handling the complexity of scheduling tasks in such complex scenarios. The behaviour of the model and its ability of generalization of the paradigm are exemplified through simulations. The results show higher flexibility for making offloading decisions.

1. Introduction

Cloud Computing paradigm is one of the most disruptive technology advances of our times. This paradigm has made Information Technology (IT) resources available to the general public through Internet. In this way, any business, organization or particular user can access to computing infrastructures and services for a fee. The progress of communication technologies has also contributed to this end. Boosting the bandwidth and hence speed of all connections has enabled us to handle more traffic. In addition, the improvements on management of cloud centres thanks to virtualization and server consolidation methods allow us to give a rapid response to changing application demands [1]. These cloud-based services evolutions have resulted in an increased outsourcing work to the cloud in all areas. This paradigm is now growing many times faster than the rest of IT industry [2].

Implementation of this paradigm to mobile computation has led to Mobile Cloud Computing (MCC) concept [3, 4]. It consists in outsourcing part of the processing load to the Cloud and getting back the results. The implications for mobile devices are evident: in the first place, they can increase their performance without any change in their hardware, and, secondly, they allow us to extend the life of battery-powered devices. These benefits have enhanced the potential of Internet of Things (IoT) paradigm and allow us to compute advanced applications on devices and embedded systems.

However, this promise of providing new resources beyond mobile computing capabilities by accessing cloud servers can lead to uncertainty and delays in response times due to unpredictability in network operations through the Internet. Further, some applications cannot expect to receive the calculated results from offloading the source. In general, there is a lack of versatility in the cloud that makes it unable to adapt to specific task requirements and to internet operation conditions. Thus, the user-perceived quality is highly variable and depends on both the application's degree of interactivity and the network's end-to-end latency. There is a requirement to improve the ability of the MCC paradigm to meet

the performance requirements for heterogeneous devices in dynamic environments where the available infrastructure can change.

From these motivation aspects exposed by these problems, the *main objective* of this work consists of researching architecture models that are able to increase the resilience and flexibility of offloading the workload outside the device and to enable scheduling the tasks along the network. To advance this goal, it is assumed as a *working hypothesis* that several network computing layers exist around the mobile device or embedded system and are available for outsourcing the application workload.

The *key contributions* of this work can be summarized as follows.

(i) A study of the complexity and key issues of distributing the processing load among several computing layers along the network and a review of the architectures and network layers to perform this offloading process.

(ii) The proposal of a general framework for a multilayer architecture to formalize the processing of an application and the implications of the distributed configurations to the performance and the communications requirements.

The *novelty* of this proposal is the extension of the MCC paradigm to the available computing layers by considering them in a comprehensive and integrated fashion with the distributed architecture.

This paper is organized as follows: in Section 2, we review most relevant models and frameworks of MCC paradigm; in Section 3 the formal framework of the computational model is introduced, and the key technical characteristics of offloading are described; Section 4 explains how the proposed multilayer computational model can handle real-world scenarios; and, finally, conclusions are drawn and some approaches for future work are also pointed out in Section 5.

2. Related Work

2.1. Outsourcing Options for MCC. The concept of MCC has been evolved in recent times with the goal of increasing the performance by offloading the workload outside the mobile devices. As a result, the outsourcing options have been increased along the communication network as shown in Figure 1.

These outsourcing options are deployed at different layers of the network forming a pool of computing platforms that are available depending on the application context.

In the top, at the network core, the *cloud layer* remains the most powerful computing platform. This is the traditional way for outsourcing the workload and takes its origin from the Cloud Computing Paradigm. Much of the current research on MCC is using this option as a default for implementing the offloading process [5–7].

Next, the *cloudlet layer* concept emerges. This infrastructure is a "little" data center with the objective of bringing the cloud closer to the users [8]. Usually, cloudlets are deployed inside the organizations that use it [9, 10]. The main

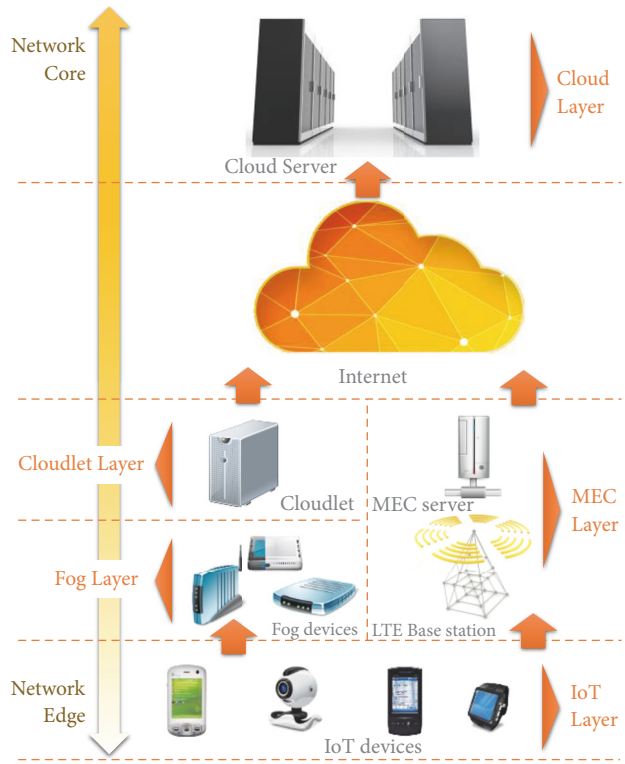

Figure 1: Outsourcing options deployed along the communication network.

advantages provided by cloudlets are twofold: in the first place, to reduce service disruptions and delays caused by the network to remote servers and secondly to ensure a high level of security of the data since it does not need to be sent through Internet [11]. Therefore, cloudlets are computing platforms similar to cloud servers but prepared and scaled according to the specific requirements of the organization.

The same principle of cloudlets is applied to the *Mobile Edge Computing (MEC) Layer*. A MEC server is a data center in a box, that, in this case, it is deployed by the mobile telecom operators in close proximity to mobile subscribers [12]. Thus, access to these computing services is made through telecommunication base stations using communication protocols such as 3G or LTE to offer a service environment with high bandwidth and low latency [13, 14]. In addition, the benefits of MEC paradigm will be enhanced by the advent of 5G communication protocol [15].

These last two layers (cloudlet and MEC) can be enabled in a dynamic way to deliver computing support to the cloud servers. In this way, they are not replacing but complementing the Cloud Computing Paradigm by providing a flexible computing power when necessary [16]. This operation encourages the development of IoT solutions and enhances the mobile applications when devices are moving over time.

Finally, a recent trend to develop the model of MCC is the Fog Computing Paradigm. The *fog layer* is a set of devices such as routers, switches, or other networking devices closer or within the end users' local networks. Their aim is to provide processing efforts as close as possible where

the data is generated [17]. Thus, these devices have some computing capabilities able to compute part of the workload of the "things" and other application devices [18, 19]. As a result, service delay is significantly reduced for end user applications.

2.2. MCC Frameworks. Realizing the vision of multilayer offloading architecture for implementing the MCC paradigm is a challenging task because of the complexity in handling the multiple aspects involved, especially those concerned with performance evaluation and tasks scheduling. In this issue, the motivation for offloading is diverse depending on the user requirements, device configuration or application constraints. Moreover, performance aspect can be of different nature such as power consumption, time delay, money cost of using external services, and network usage.

There is a need to design suitable frameworks to formalize the performance components in a homogeneous way to make decisions on when and where to outsource the processing load. In this subsection, a review of recent frameworks of MCC paradigm is described in order to reach the knowledge border in this issue.

There are several existing frameworks and architectures in the literature for outsourcing the tasks of the application workload to other computing platforms. The areas of application of these techniques cover many sectors and disciplines. In general, they develop the IoT and Cyber-Physical Systems (CPS) paradigms that are experiencing a great growth in recent years. The devices involved are a heterogeneous set which mainly consists of sensors, actuators, and embedded systems. In addition, the mobile phones and other mobile devices such as wearables are also in this kind of applications.

As overview of the MCC proposals, they aim to carry out a collaborative work for distributing the processing load and meeting the application requirements [20, 21]. In most cases, these applications have special features and are high demanding of computing resources such as multimedia analysis [22], high volume of data [23], or real-time constraint [4]. By distributing the work, the cloud architecture bottleneck is overcome, and better delay and predictability can be achieved [24, 25].

In general terms, offloading from the "things" or mobile devices to cloud, edge, or cloudlet server always produced significant increase in performance. It is clear that the slower the devices are, the sharpest these differences become. However, the offloading criteria can consider other aspects such as network usage or money cost of external computing services [26]. In these cases, an indirect relation exists between using outsourcing services and benefits. Thus, the benefit function needs to be redefined to consider a heterogeneous type of aspects in order to make decisions on where and when to outsource the application workload.

The key parts of a framework for handling the offloading process are the architecture that defines the available outsourcing platforms, the decision method on where and when to offload, and the communication model that defines how to perform this process.

Regarding the decision method for offloading, in general term, it is considered as a scheduler which decides when and where to offload taking into account the application constraints, the tasks' features, and the performance aspects of the available computing platforms. Its main aim is how to reasonably allocate the tasks to available computing platforms to minimize the total cost and load. The optimal scheduling scheme belongs to NP-complete problem set and, therefore, traditional strategies cannot be applied in a suitable way for MCC applications.

To address this issue, many approaches have been proposed based on different mathematical techniques and algorithms. In recent approaches, the scheduling method can be formulated as a constrained optimization problem where a suboptimal solution is usually the best choice. These methods can be used for multitask [27] and multiobjective [28] scheduling problems in several scenarios. In the same line, there are methods based on stochastic techniques and search algorithms to deal with the high complexity of this problem [29, 30]. Other proposals are based on dynamic programming methods in order to handle the runtime variations of tasks execution [31]. Also, a decision tree can be built to classify the conditions and decision parameters in order to get a fast offloading response [32]. Finally, other works propose innovative approaches to address this problem based on techniques used for processor scheduling area such as genetic algorithms [33], and imprecise computation strategies [4].

In most cases, the offloading decisions are mainly focused on energy optimization [31] since it is a critical issue for mobile devices. However, other performance aspects such as response time and Quality of Service (QoS) are also being considered by these methods [4, 27, 29, 33].

The frameworks can be implemented as a set of procedures, methods, and tools. These components can be part of the device itself or installed in a middleware layer in an external supervising device [34]. This layer is introduced as an intermediary between the devices and the computing platforms and it works as a smart gateway that monitors the underlying nodes and decides if offloading is needed or not [35, 36]. Table 1 summarizes the recent proposals on these aspects.

2.3. Findings. After reviewing the frameworks for offloading mobile computation, three main findings can be drawn that justify our proposed model for designing multilayer MCC architectures:

(1) The general objective of existing MCC architectural approaches is to improve the overall application functioning. However, in most cases, they do not consider multiple options for offloading the work nor the intermediate network infrastructure.

(2) There are numerous frameworks on how to distribute the computation of the applications to perform partial remote execution. They are mainly focused on minimizing the energy consumption of devices, increasing the performance, and maximizing the overall QoS. However, they do not consider heterogeneous performance metrics such as money cost or global network traffic.

Table 1: Recent frameworks for MCC.

Work	Key aspects of framework
Collaborative Working Architecture [20]	General architecture and scheduler. *OT: ad hoc cloud, fog, and Cloud*
Multilayered scheduling [37]	Tasks specification, scheduling method. *OT: ad hoc cloud of CPS nodes*
Cuckoo [38]	Programming model and integration tools. *OT: Cloud*
Data storage framework [39]	Architecture, Database Management, and File Repository Model. *OT: Cloud*
Flexible Framework [4]	General architecture and Scheduling method. *OT: Cloud*
Cyber-Manufacturing [40]	Architecture, communication protocol and analysis. *OT: Cloud, cloudlet, fog*
Federated IoT services [41]	Management, problem formulation, and heuristic for tasks allocation over 5G. *OT: MEC, ad hoc clouds*
VM migration framework [42]	Smart precopy live migration approach. *OT: Fog, Cloud*
Cloudlet in MCC [10]	Architecture, Stochastic performance modelling. *OT: Cloudlet, Cloud*
Resource usage optimization [5]	Architecture, Resource usage, and performance evaluation modelling. *OT: Cloud*
Mobile code offloading [43]	Architecture, offloading methods in Java and push notifications. *OT: Cloud*
QoS Aware Computation Offloading [44]	Problem formulation, optimal offloading decision process. *OT: ad hoc cloud*
Adaptive MCC framework [45]	Application partitioning, offloading decision algorithm. *OT: Cloud*
Edge Computing Framework [22]	Communication and Computation Models. *OT: Edge nodes*
Distributed computational model [46]	Resource utilization specification, management system. OT: CPS nodes, Cloud
Scheduling internet of things applications [47]	Scheduling method, performance metrics. *OT: Cloud*
IoT and Cloud Computing Integration [48]	Architecture and components. *OT: Cloud*
Context-aware computation offloading [49]	Design pattern and estimation model. *OT: Cloud*
Edge-Fog cloud [50]	Method for distributing the processing tasks. OT: Edge and fog nodes
Framework for code offloading [25]	Architecture and offloading decision-making engine. *OT: Edge and Cloud*

Offloading target (OT).

(3) The existing works in the literature do not provide a formal framework to formalize the overall offloading process; rather, they conduct numerical results of their reference architectures. Other works have insufficient depth, or they are focused on specific issues of each layer separately.

The research presented in this article pursues the same objectives as mentioned above. To the best of our knowledge, this is the first study to provide, from a holistic perspective, a comprehensive model that provides an analysis of technical aspects and includes all known computing layers of the network. This model will allow us to handle complexity of the offloading process of MCC systems.

Next section introduces the proposed multilayer architecture and presents a formal basis for modelling the offloading process in the MCC paradigm.

3. Multilayer Architecture Model

3.1. Proposed Distributed Architecture. This architecture design promotes an adaptation to changing environments and enables a dynamic scaling of computational power able to assume a variable and/or intensive application workload in a more effective manner than existing proposals. The network layers considered for offloading computation can be those described in Section 2.1. An overview of the network layers to be used by the proposed framework is depicted in Figure 1.

This approach attempts to identify methods to obtain the best results and performance using the network infrastructure deployments and local processing capabilities. Multiple design configurations can be supported. The mobile devices and connected "things" can be heterogeneous and have different capabilities for processing data. The middle infrastructure layers can be deployed by stakeholders to improve the execution of their mobile or IoT applications and extend them to more potential customers, for example, advanced multimedia games and complex financial apps. These layers can be equipped with specialized hardware components for improving the complex calculations, including custom cloudlets with GPUs, DSP, and cryptographic coprocessors.

Below, the main technical characteristics of the MCC paradigm are introduced together with the implications and advantages provided for them by the proposed model.

The *application partitioning method* is a critical aspect for enabling the outsourcing of the code at multiple heterogeneous layers of the network architecture, even at the nearest available layers. In this manner, although an elastic approach is desirable to provide flexibility, the time determining the offloadable code must be minimized. Thus, the proposed architecture must use a static partitioning to know, at any step, the candidate parts of the application to be offloaded. Other dynamic proposals can be applicable in a complementary fashion if resources exist.

The *offloading decision mechanism* can be, at the first layer for either of the mentioned options: static or dynamic.

Table 2: Technical characteristics of offloading methods and frameworks.

MCC technical characteristics	Methods	Key aspects for multilayer architecture
Application partitioning method	Static, dynamic	Exploit the available nearest resources
Offloading decision mechanism	Static, dynamic	Leverage the potential of the available resources
Granularity unit	Module, bundle, subroutine, process, thread, class, component, method	Provide flexibility to the application needs
Offloading mode	Client-server communication, virtualization, mobile agents	Support heterogeneous infrastructures
Context-awareness offloading	Adaptive schemes	Awareness of available layers
Cost-benefit model	Resource monitoring and profiling, parametric analysis, stochastic methods	Decide where to perform the computations

However, to ensure that the maximum advantage of the availability of multiple layers with different profiles is selected, the dynamic option is preferable. In this case, the architecture must calculate, for each layer, the offloading decision according to an optimization problem that addresses all the issues specified. This algorithm must be able to be executed quickly to leverage the potential of the available resources. In the previous section the latest trend on this issue can be found. In any case, fast methods [32] and real-time strategies [4] must be introduced to provide a rapid response and address the time constraints.

The desirable *granularity unit for outsourcing* must be as small as possible to provide the highest flexibility; however, small size implies higher management cost. The granularity unit for the architecture can be variable, depending on the features of the target platform for offloading. That is, small parts of the application can be outsourced for fast execution on surrounding platforms and other intensive parts can be offloaded to specialized platforms. The optimal partitioning is an NP-complete problem [51]. To avoid time consumption in the automatic analysis of the code, the variable granularity approach can be made in the design stage of the application. This requires the collaboration of the application developers by annotating into the code the possibilities for outsourcing.

The *offloading mode* for the proposed MCC architecture must meet the following minimum requirements: the method must be suitable for heterogeneous infrastructures and must use a lightweight method so as to not overload the limited resources of the edge layers (devices of ad hoc clouds and fog nodes). The mode based on mobile code would appear to be the best option; however, it requires further research on mobile agent management to become a robust option for implementing heterogeneous MCC architectures [3]. Until then, a combination of virtualization and client-server methods offers the best results for each layer.

The *context-awareness feature* is important to perform offloading to several layers of the network. The desirable configuration requires a self-adaptive architecture to know what layers and devices are available at any time for offloading the work. It is significant to perform this feature without intervention of an administrator to shape dynamic environments. It is precisely in these complex environments where many options exist and where the proposed architecture attains its full potential.

Finally, the *cost-benefit function* must be computed on each level of the architecture to determine where to perform the computation. Thus, complex algorithms are not suitable for this function, although suboptimal decisions are found. A stability of the performance for network infrastructure in the same working conditions is supposed. Hence, prediction models based on look-up tables of history data are a fast approach to the offloading of application work. Further, this information can be complemented at minimum cost with data from the device itself and the main network parameters.

Table 2 presents the technical characteristics of offloading methods and frameworks for the MCC paradigm and highlights the key aspects for a multilayer architecture.

The descriptions and recommendations regarding the operation of the proposed architecture reveal that there continue to be important challenges that must be addressed to leverage the available infrastructure at multiple layers of the network. In this regard, this work introduces basic ideas and notes on specific research issues for implementing a multilayer architecture.

The next subsection describes the formal framework of the architecture and the elements involved in the distributed computing.

3.2. General Framework. This subsection describes the general aspects of the multilayer distributed architecture for outsourcing the processing load using the combination of computing layers of the available infrastructure.

According to the stated working hypothesis, the main idea behind the multilayer architecture to address computation offloading is to offer a set of options for performing the computations at different network layers where available infrastructure exists. As a rule, it is advisable to perform the processing as close as possible to where the data are acquired to reduce the delay and global network traffic. However, the final decision on where to offload each task depends on many other aspects including application requirements, devices configuration, user preferences, size of input data, and pricing. The result is a flexible and scalable model where the computations can be performed on a variety of platforms and computing layers. The formulation introduced in this subsection is utilized to better describe the contributions of the proposed architecture for providing flexibility to the processing requirements of IoT applications. Important

Table 3: Summary of Notations.

Notation	Definition (#Expression)
t_j	Application Task
$\Gamma_{Appl} = \{t_1, t_2, \ldots, t_n\}$	Set of Application tasks (1)
$\mathscr{L} = \{L_0, \ldots, L_{cc}\}$	Set of available computing layers (2)
p_j	Computing platform
p_{kj}	Computing platform of layer k
p_{0j}	Computing platform of layer base
$L_j = \{p_{j1}, p_{j2}, \ldots, p_{jmj}\}$	Set of computing platf. of layer j (3)
$A_{p0i}^{time} = \{p_{1,i}, p_{2,i}, \ldots, p_{cc,i}\}$	Set of the available upper platforms of device p_{0i} at a *time* (4)
$\Omega = \{\text{delay}, \text{power}, \ldots\}$	Set of performance aspects (5)
$\wedge(\Gamma_{Appl}) = \langle (t_1, p_{k1,j1}), \ldots \rangle$	Sequence of platforms on which the application is processed (13)
$\wedge_{min}(Cmp_{\alpha}^{time})$	List of platforms that meets the minimum computing costs for the α performance aspect at a given time
$\wedge_{min}(Net_{\alpha}^{time})$	List of platforms that meet the minimum communication costs for the α performance aspect at a given time
Δ	Saturation value in calculation cost
$\text{Data}(t_i,)$	Volume of data generated by task t_i (10)
$\text{Data}(t_i, p_{j,k})$	Necessary data to be moved for computing the task t_i in the platform p_j of the layer k (11)

Table 4: Summary of Formulations.

Formulation	Definition (#Expression)
$Cmp_{\alpha}^{time}(t_i, p_{k,j})$	Processing cost of the task t_i in the platform $p_{k,j}$ for the α performance aspect (7)
$Cmp_{\alpha}^{time}\left(\Gamma_{Appl}\right)_{min}$	Minimum processing cost of the application for the α performance aspect (17)
$Net_{\alpha}^{time}(t_i, p_{k,j})$	Communication cost to move the task t_i to the platform $p_{k,j}$ for the α performance aspect (9)
$Net_{\alpha}^{time}\left(\Gamma_{Appl}\right)_{min}$	Minimum communication cost of the execution of the application for the α performance aspect (18)
$E_{\alpha}^{time}(\Gamma_{Appl})$	Execution cost of the application for the α performance aspect (6), (14)
$E_{\alpha}^{time}\left(\Gamma_{Appl}\right)_{min}$	Minimum execution cost of the application for the α performance aspect (16)
$Net_{\alpha}^{time}\left(\Gamma_{Appl}\right)\wedge_{min}$	Minimum communication cost through the distributed architecture for the minimum processing cost (19)
$\text{Data}(\Gamma_{Appl})$	Volume of data to be moved for computing the whole application (15)
$\text{Data}\left(\Gamma_{Appl}\right)_{min}$	Volume of data to be moved through the distributed architecture to run the application within the minimum cost. (20)

notations and expression used in this paper are provided in Tables 3 and 4.

First, we consider a granularity unit for offloading the application task. This unit can be one of those mentioned in Table 2 under the condition that the applications considered by this framework consist of a list of tasks. The tasks can be executed sequentially or in parallel depending on the interrelationships between inputs and outputs. These tasks can be variable sized depending on the offloadable code. For example, a task can be a fragment of code, such as a set of instructions, a process, a method, or a subroutine.

Let Γ_{Appl} be the set of tasks of an application:

$$\Gamma_{Appl} = \{t_1, t_2, \ldots, t_n\}. \quad (1)$$

These tasks can be processed at different computing layers and platforms. Let $\mathscr{L}$ be the set of available computing layers of the architecture:

$$\mathscr{L} = \{L_0, \ldots, L_{cc}\} \quad (2)$$

The number of network processing layers depends on numerous aspects such as execution environment, available infrastructure, and configuration options. In all these cases, L_0 is the layer of the single device where the task is generated and L_{cc} is the remote cloud-computing server. In between can be several available computing layers to perform the processing at different levels. For example, in an autonomous vehicle application context, L_1 can be the network composed of several nearby vehicles, L_2 can be the layer formed by the city traffic infrastructure or deployed cloudlets, and L_3 can be the mobile edge-computing infrastructure behind the communication network. Other configurations of the infrastructure can be configured for this application.

Each layer has a set of computing platforms, which can be heterogeneous with different processing capabilities and abilities according to their characteristics. Thus, layer L_j has m_j computing platforms, where $L_j \in \{0, \ldots, cc\}$ and $m_j > 0$:

$$L_j = \left\{p_{j1}, p_{j2}, \ldots, p_{jmj}\right\}, \quad (3)$$

where p_{jk} is platform k of layer j.

The different layers of the network can be deployed in sequence or in a parallel configuration where each computing platform of a layer can execute the services of the upper layers and provides services to several elements of the lower layers.

For a specific device (p_{0i}), the list of available upper platforms at a given *time* can be expressed by A_{p0i}^{time}:

$$A_{p0i}^{time} = \{p_{1,i}, p_{2,i}, \ldots, p_{cc,i}\} \quad (4)$$

The list of platforms described in Expression (4) represents the specific network architecture for the device and indicates the possibilities for offloading the work at the given time. This list can vary with time depending on the application context. For example, if the mobile device is moving, the available infrastructure can change.

Several expressions can be introduced into the proposed general framework to define the behaviour of the multilayer architecture related to performance and resource consumption issues. In this section, the execution cost and data communication are analysed. The execution cost can be any of the different performance aspects that are involved in the execution of a task in a device. In this manner, let Ω be the set of performance aspects to consider:

$$\Omega = \{\text{time delay}, \text{power consumption}, \text{money}, \ldots\} \quad (5)$$

For each of these aspects, the overall execution cost (E) of an application in this distributed infrastructure consists of two main components as indicated in Expression (6): (a) the computing cost (Cmp) and (b) the communication cost along the network and layers (Net):

$$E_{\alpha}^{time}\left(\Gamma_{\text{Appl}}\right) = Cmp_{\alpha}^{time}\left(\Gamma_{\text{Appl}}\right) + Net_{\alpha}^{time}\left(\Gamma_{\text{Appl}}\right), \quad (6)$$

where $\alpha \in \Omega$.

The cost expressions are a function of time because the execution conditions can change at any time depending on the workload currently processing on each platform, the other processes that eventually are executing simultaneously, and the network traffic situations.

Related to the first aspect (a), the processing capability of each computing platform can be in a range from zero to extremely high. Further, there may be platforms with specific capabilities that provide services to many applications and allow the acceleration of the processing of specific types of tasks. For example, GPUs can be installed on the cloudlet servers to accelerate multimedia algorithms. In this manner, the granularity of this calculation is the cost of computing each task. For each task t_i of the application, the cost of executing t_i on each element of the distributed architecture can be determined. Expression (7) is the cost of processing the t_i task in platform p_j of layer k for the α performance aspect:

$$Cmp_{\alpha}^{time}\left(t_i, p_{k,j}\right) \in \mathbb{R}^{+} \bigcup \{0\} \quad (7)$$

The workload of a platform can vary during the day depending on the number of devices simultaneously connected and other features. This fact is common in the intermediate platforms and in the cloud infrastructure because they collect data from different elements of the lower levels. However, it is normal that they have redundant computing elements that perform massive parallel processing. If a platform cannot compute a task, it will be assigned a cost of the saturation value δ. That is,

$$\begin{aligned} &\forall x \in \mathbb{R}^{+} \bigcup \{0\}, \\ &x + \delta = \delta. \end{aligned} \quad (8)$$

In addition to computational cost, the framework considers the communication cost along the architecture, that is, the cost of moving the tasks between the platforms and layers as well as the data they require. The following expression indicates the communication cost to move task t_i to platform p_j of layer k for the α performance aspect:

$$Net_{\alpha}^{time}\left(t_i, p_{k,j}\right) \in \mathbb{R}^{+} \bigcup \{0\} \quad (9)$$

This expression is also a function of time and can have a variable result at any time according to different aspects such as network congestion, device connectivity, bandwidth availability, and pricing. The cost of moving the data in the same processing element is null; that is, executing the entire application on the same platform processor has no communication cost. Further, as in the case of the computational cost, if it is not possible to move the data to the p platform, then the communication cost is δ, for example, when the platform is not connected to the place where this data was generated or there is no connectivity.

The communication cost can be any performance aspect of the set Ω defined in Expression (5). A complementary expression can be defined to determine the transferred data (in data units) between platforms when the processing is distributed. Expressions (10) and (11) indicate, respectively, the volume of data generated by each task and the necessary data to be moved for computing task t_i in platform p_j of layer k. If the task is processed in the same platform, the necessary data is zero. These functions are independent from time:

$$\text{Data}\left(t_i, \right) \in \mathbb{R}^{+} \bigcup \{0\} \quad (10)$$

$$\text{Data}\left(t_i, p_{j,k}\right) \in \mathbb{R}^{+} \bigcup \{0\} \quad (11)$$

Normally, the necessary input data of a task corresponds with the generated output data from the previous task:

$$\text{Data}\left(t_{i+1}, p_{j,k}\right) = \text{Data}\left(t_i, \right) \quad (12)$$

The data flow and connectivity of the computing platforms define a graph to share and distribute the application workload. In the base of the graph are the mobile/embedded devices; the upper side is formed by the cloud-computing servers. Between these two sides, several intermediate computing platforms can be installed. This infrastructure allows executing advanced applications and provides improved overall performance.

In this architecture, movements of the tasks can occur based on the offloading configuration and execution costs. The many possible options allow flexible implementation of the applications to optimize any of the system parameters including minimizing response time, reducing the data flow through the communication network, minimizing the energy consumption of the devices, increasing the processing time of the cloud system, and minimizing the money cost of using external resources. Hence, the proposed multilayer architecture allows the design of numerous configurations for the execution of tasks depending on the type of application, execution restrictions, or operating conditions considering the above aspects.

The sequence of platforms on which the application executes is defined by the vector $\wedge(\Gamma_{\text{Appl}})$ as follows:

$$\wedge\left(\Gamma_{\text{Appl}}\right) \equiv \left\langle \left(t_1, p_{k1,j1}\right), \ldots, \left(t_n, p_{kn,jn}\right) \right\rangle, \quad (13)$$

where $\forall t_i \in \Gamma_{\text{Appl.}}\ \wedge_{i[k,j]} = (t_i, p_{k,j})$; that is, task t_i is processed on platform j of layer k.

Then, the execution cost of an application can be obtained expanding Expression (6) with the platforms of the vector $\wedge$:

$$E_{\alpha}^{time}\left(\Gamma_{\text{Appl}}\right) = \sum_{\text{i}} \left[Cmp_{\alpha}^{time}\left(\wedge_{\text{i[k,j]}}\right) + Net_{\alpha}^{time}\left(\wedge_{\text{i[k,j]}}\right)\right]. \quad (14)$$

Similarly, the amount of data to be moved for computing the entire application is obtained from the next expression:

$$\text{Data}\left(\Gamma_{\text{Appl}}\right) = \sum_{\text{i}} \text{Data}\left(\wedge_{\text{i[k,j]}}\right). \quad (15)$$

From the previous expressions, calculations can be made to optimize the processing according to the configuration criteria of the architecture. Consequently, a more appropriate sequence of platforms for outsourcing is obtained. This information can guide the scheduling methods and the offloading strategy to achieve the best performance.

The following expression obtains the minimum cost of the execution of the application considering the cost of the platforms and cost of data movement along the communication network:

$$E_{\alpha}^{time}\left(\Gamma_{\text{Appl}}\right)_{\text{min}} = \min_{\text{k,j}} \left\{\sum_{\text{i}} \left[Cmp_{\alpha}^{time}\left(\wedge_{\text{i[k,j]}}\right) + Net_{\alpha}^{time}\left(\wedge_{\text{i[k,j]}}\right)\right]\right\} \quad (16)$$

The cost of one of the components of Expression (6) could be similarly obtained. That is, the next expressions produce the cost for the separate components:

$$Cmp_{\alpha}^{time}\left(\Gamma_{\text{Appl}}\right)_{\text{min}} = \min_{\text{k,j}} \left\{\sum_{\text{i}} Cmp_{\alpha}^{time}\left(\wedge_{\text{i[k,j]}}\right)\right\} \quad (17)$$

$$Net_{\alpha}^{time}\left(\Gamma_{\text{Appl}}\right)_{\text{min}} = \min_{\text{k,j}} \left\{\sum_{\text{i}} \left[\begin{array}{c}\text{if } Cmp_{\alpha}^{time}\left(\wedge_{\text{i[k,j]}}\right) < \delta\text{: } \text{Net}_{\text{time}}\left(\wedge_{\text{i[k,j]}}\right) \\ \text{else: } \delta\end{array}\right]\right\} \quad (18)$$

Note that Expressions (16), (17), and (18) only consider the calculations for the platforms which are able to compute the tasks.

The list of platforms that meets the minimum cost is defined as follows. Let $\wedge_{min}(Cmp_{\alpha}^{time})$ be the sequence of platforms executing each of the tasks that achieve the minimum computing cost of the application and let $\wedge_{min}(Net_{\alpha}^{time})$ be the sequence of platforms that achieve the minimum communication cost through the distributed architecture among the different options for offloading.

Then, considering the minimum processing cost for one performance aspect α (for example time delay), the following expression obtains the minimum communication cost through the distributed architecture to execute the application:

$$Net_{\alpha}^{time}\left(\Gamma_{\text{Appl}}\right)\wedge_{\text{min}} = \sum_{i} Net_{\alpha}^{time}\left(\wedge_{\text{i[k,j]}}\right), \quad (19)$$

where $\wedge_{\text{i[k,j]}} \in \wedge_{min}(Cmp_{\alpha}^{time})$.

Finally, the following expression obtains the amount of data to be moved through the distributed architecture to execute the application within the minimum cost:

$$\text{Data}\left(\Gamma_{\text{Appl}}\right)_{\text{min}} = \sum_{\text{i}} \text{Data}\left(\wedge_{\text{i[k,j]}}\right), \quad (20)$$

where $\wedge_{\text{i[k,j]}} \in \wedge_{min}(Net_{\alpha}^{time})$.

It is notable that the information regarding computational and communication costs is not fixed and it is time dependent for some of the performance aspects. Thus, the computing platforms of the architecture can follow different strategies to derive the correct decisions regarding offloading tasks to another platform or layer.

First, it can use prediction techniques based on historical data. In this manner, the devices can know the estimated performance of the available platforms quickly. A possible implementation can be a look-up table that stores the performance data for each interesting aspect. The tasks can be clustered to ensure manageable table sizes. For example, the similarity criteria can be any indicator of the type of specialized processing required, such as integer, floating point, multimedia, or cryptographic. This feature should be noted in the programming stage of the application to ease the offloading process. Moreover, this data must be updated after each operation.

Secondly, there are methods for periodically testing the data time costs. Light threads can be launched at the beginning of processing to request performance conditions of the architecture.

A combination of the above methods can be made to obtain more accurate information regarding the execution context of the architecture.

In any case, an embedded middleware layer could be necessary to take charge of this job. This layer is already playing an increasingly important role in the edge computing paradigm to perform discovery and other broker services [52]. In this case, it is responsible for gathering and evaluating all the relevant data and identifying the available options to make the right offloading decision. The cost of this decentralized approach is lower than a centralized control hosted somewhere along the network. However, the rational operation of each device in *ad hoc* scenario should be satisfactory, but presumably suboptimal [53].

4. Application Examples

This section describes how the proposed multilayer computational model can handle real-world scenarios in order to find the best execution cost ($E_{\alpha}^{time}(\Gamma_{\text{Appl}})$) for computing an application taking into account the different available offloading layers. For this purpose, three application contexts have been defined to test the model under different conditions. These scenarios correspond to operation modes affected by user preferences or device configurations. This section just shows three examples. Of course, other application contexts could be defined. In all cases, the main contribution of the proposed model is the versatility in formalizing the offloading requirements of each environment and the better leveraging of the available resources in the communication network. The

exact values in simulation and numerical examples provided in this section do not impact essentially to these outcomes.

In these scenarios, the IoT layer is composed by mobile devices such as smartphones, tablet PCs, and smartwatches. These are heterogeneous devices and might have different computing and communication capabilities. In addition, the users might configure the devices according to their preferences and economic plans.

The example application consists of an Augmented Reality (AR) system for Smart Cities which enables the user to move freely through the modelled environment of the city using their mobile devices. This technology recognizes what you are doing and then enhances it. The AR systems for Smart Cities have great potential for all involved [54], for example, the disabled citizens in order to know the resources for inclusive city and accessible paths along the city. Moreover, the latest AR solutions consider edge resources together with cloudlets and cloud server as high-end computing platforms to offload the AR applications [55]. Therefore, the available layers of the architecture are $\mathscr{L}_A$ = {IoT devices, fog computing, fog nodes, cloudlet, MEC server, Cloud server}.

In this example, the scheduling algorithm of the middleware could be based on prediction techniques and, for example, the performance estimations could be like those shown in Table 5 for each case. These data have been retrieved from other sources and on our own experiments of previous research on this matter [4, 46]. Scheduling algorithms based on historical data are common in dynamic environments [56]. Therefore, these performance estimations are recorded in a Look-Up Table that can be embedded in each device and frequently updated by broker services.

4.1. Battery Saving Application Context. The general definition of these contexts is that the user device is configured to save battery power. Battery consumption is a performance aspect for only battery-powered devices such as mobile devices or wireless sensors. Thus, it only applies to IoT devices. In addition, this is a static feature. That is

$$\forall k > 0, \quad Cmp^{time}_{batt.cons.}\left(t_i, p_{k,j}\right) = 0 \tag{21}$$

It means that the processing cost of outsourcing platforms does not matter for this configuration since the only important thing is energy saving. Thus, under this configuration, the application workload will be outsourced whenever and wherever possible. However, the communication costs for moving the tasks to another computing platform are not void since data communication consumes battery. Therefore, in this scenario, the computation of a task (t_i) will be offloaded when battery consumption of local execution is greater than communication cost to some outsourcing platform ($p_{k,j}$). That is

$$\exists k > 0, \quad Net^{time}_{batt.cons.}\left(t_i, p_{k,j}\right) < Cmp^{time}_{batt.cons.}\left(t_i, p_{0,j}\right) \tag{22}$$

This consumption depends on where the data is moved. Generally, communication costs through a wireless local network are lower than through a telecommunication network such as LTE [57]. Therefore, in this case a table with the communication cost of the mobile device is needed to implement the Net function. For example, Table 5(A) shows estimated data of battery consumption ($Net^{time}_{batt.cons.}(t_i, p_{k,j})$) for different communication options.

As can be seen in Table 5(A), the communication cost through mobile operators is higher than through wired Internet. For this reason, the cost of cloud server is different according to how the connection is made.

4.2. Monetary Cost Saving Application Context. The general definition of this context is that the user device does not have performance enough to run the VR application. It is only used for displaying the results and, therefore, it has to outsource the processing load. In such a context, it is configured to save processing expenses when using an external computing platform.

This is a similar approach than the previous one but, in this case, the device is unable to compute any task; that is

$$Cmp^{time}_{money}\left(t_i, p_{0,j}\right) = \delta \tag{23}$$

This scenario supposes that using outsourcing resources has money cost under the *Infrastructure-as-a-service* (IaaS) model. That is, the workload can be outsourced, but the infrastructure owner will charge you the cost. In addition, the cost can be variable depending on where the computing platform is placed, its performance, and the moment (hour/day/month) when it is required. The new cloud service brokerage efficiently trades infrastructure cloud services among multiple resource providers and consumers [58, 59]. In this way, billing is completely flexible and just based on resource usage.

However, in order to know the processing cost, a dynamic monitoring is needed. For example, Table 5(B) shows an estimated data ($Cmp^{time}_{money.}(t_i, p_{k,j})$) for the different outsourcing options at an instant *time*. In this case, fog layer is not considered since it is difficult to estimate the monetary cost of outsourcing to it. In addition, due to its limited computing capabilities, this layer is usually used for short periods of time which are not comparable with published hourly rates.

Generally, cloudlet and MEC price should be higher than cloud since they cannot take advantage of economies of scale. In addition, they are limited to a more restricted area, and thus, they have less competitors for outsourcing. From other points of view, cloudlet layer can be deployed and owned by organizations and therefore they should have no costs for their users.

Regarding Net function, the devices might have a communication cost. It could also be taken into account in order to offload through local wireless or communication network, since, in many cases, the former has no costs.

4.3. Real-Time Application Context. The general definition of these contexts is to minimize the computing delay of the application in order to meet quality constraints of demanding real-time AR algorithms.

Table 5: Performance estimations.

Offloading target	Application context			
	(A) BATTERY SAVING	(B) MONETARY COST	(C) REAL-TIME	
	Ω = {battery consumption}	Ω = {money}	Ω = {time delay}	
	$Net^{time}_{batt.cons.}(t_i, p_{k,j})$(mJ/Kbps)[a]	$Cmp^{time}_{money.}(t_i, p_{k,j})$ ($/hour)	$Net^{time}_{timedelay}(t_i, p_{k,j})$ (ms/KB)	$Cmp^{time}_{timedelay}(t_i, p_{k,j})$ (ms)
Fog nodes	1		0.2143	250
Cloudlet	1	0.256	0.4286	52
MEC server	1.56	0.128	0.7812	52
Cloud server	1.56[b] / 1[c]	0.064[d]	2.2177	50

[a]Estimations from [57].
[b]Cloud server connection through mobile station.
[c]Cloud server connection through cloudlet.
[d]On-demand instance for *t2.medium* server (https://aws.amazon.com/ec2/spot/pricing/).

In this scenario, there exists a very dynamic nature of the cost matrix since it depends on the current workload of each computer. In addition, the data must be moved to the target computing platform, and therefore the communication costs are responsible for a relevant part of the total delay [45, 60]. For example, Table 5(C) shows an estimated data at an instant *time*.

As can be noted from the above data, the communication costs are increasing with distance from outsourcing platforms. In addition, fog and cloudlet layers can be into the same Local Area Network (LAN) of the device or at a very close one. Of course, these costs depend on communication technology used. For 5G technology the delay of MEC and cloud platforms will be significantly decreased.

Regarding computing delay, in general, the computing costs of intermediate computing platforms and the cloud server are significantly lower than those of the device itself. In this way, a decreasing computing cost must occur for the most of application tasks:

$$\forall p_{k,j} \in A^{time}_{p0i},$$
$$Cmp^{time}_{\alpha}\left(t_i, p_{k,j}\right) < Cmp^{time}_{\alpha}\left(t_i, p_{k-1,j}\right) \quad (24)$$

This consideration is quite frequent in environments designed for outsourcing of tasks from mobile devices or connected "things."

Table 5(C) shows the fact that fog nodes are limited resourced platforms, but useful for reaching a fast response in data operation. Another aspect taken into account is that cloud servers are powerful platforms, but they are usually public infrastructures used by many applications at once. Thus, their perceived performance is similar to MEC and cloudlet platforms.

There may be certain applications more suited than others to make the most of this model. Certainly, applications containing intensive computing tasks will be more candidates to outsource than others; moreover, those tasks that require or produce a large volume of data should be processed as close as possible to the data sources to minimize communication costs.

5. Conclusions and Future Work

MCC is a recent paradigm with disruptive implications for mobile computing and for the development of IoT and CPS advanced applications. This paradigm promotes the QoS of devices and 'things' by outsourcing their computation tasks to external computing platforms. The majority of proposals in MCC take into account only the cloud layer for outsourcing the application workload. Another computing infrastructure hosted at different network layer is also being introduced recently (cloudlets, fog nodes, etc.).

This work generalizes the idea of MCC paradigm to multiple computing platforms at different network layers. The proposed model considers not only the cloud layer, but also other network computing layers such as fog computing, Mobile Edge Computing, and cloudlet computing platforms.

To the best of our knowledge, this paper is the first work that extends the MCC paradigm to the available computing layers by considering them in a comprehensive and integrated fashion with the distributed architecture. To this end, a general framework of a multilayer network architecture is described. The proposed model formalizes the processing of the application workload and the implications of the distributed configurations in the performance and in the communications needs. This model offers a versatile approach where different performance aspects can be considered within the same framework (time delay, power consumption, money, etc.). In this way, the proposal analyzes from a holistic perspective the technical aspects involved in Mobile Cloud Computing paradigm taking into account the contributions and results of the more recent works on these topics.

This framework enables decision making regarding scheduling the outsourcing of the applications tasks based on the current and historical information of the systems. Several examples of these features have been described in three scenarios where different performance aspects and computing platforms are involved. Each of them is conditioned by its user preferences or device configurations. The results show the versatility of the model to represent all the elements involved.

In the future, this research can be extended to cover remaining challenges around this paradigm, for example, (a)

the specification of the network topology and link capacity of the computing platforms along the network; (b) the definition of a middleware layer to drive the outsourcing process; (c) the introduction of discovery and broker services, and a suitable decision method for outsourcing; and (d) the design of mechanisms for collecting and disseminating the performance information, and the evaluation about the associate cost for doing that. In any case, the extension of the model can be made from the proposal of this work by adding new modules and features.

Data Availability

The available data can be found in the references and web pages cited in the document.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Spanish Research Agency (AEI) and the European Regional Development Fund (ERDF), under Project CloudDriver4Industry TIN2017-89266-R, and by the Conselleria de Educación, Investigación, Cultura y Deporte, of the Community of Valencia, Spain, within the program of support for research under Project AICO/2017/134.

References

[1] K. Li, "Optimal temporal partitioning of a multicore server processor for virtual machine allocation," *IEEE Access*, vol. 6, pp. 54726–54738, 2018.

[2] IDC, International Data Corporation's Worldwide Semiannual Public Cloud Services Spending Guide,2018, Available online at: https://www.idc.com/getdoc.jsp?containerId=IDC_P33214.

[3] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, 2013.

[4] H. Mora, D. Gil, J.-F. Colom-López, and M. T. Signes-Pont, "Flexible framework for real-time embedded systems based on mobile cloud computing paradigm," *Mobile Information Systems*, vol. 2015, Article ID 652462, p. 14, 2015.

[5] P. Nawrocki and W. Reszelewski, "Resource usage optimization in Mobile Cloud Computing," *Computer Communications*, vol. 99, pp. 1–12, 2017.

[6] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: issues and challenges," *Applied Computing and Informatics*, vol. 14, no. 1, pp. 1–16, 2018.

[7] V. Haghighi and N. S. Moayedian, "An offloading strategy in mobile cloud computing considering energy and delay constraints," *IEEE Access*, vol. 6, pp. 11849–11861, 2018.

[8] M. Satyanarayanan, "Mobile computing: the next decade," in *Proceedings of the ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, 2010.

[9] U. Shaukat, E. Ahmed, Z. Anwar, and F. Xia, "Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges," *Journal of Network and Computer Applications*, vol. 62, pp. 18–40, 2016.

[10] H. Raei, N. Yazdani, and R. Shojaee, "Modeling and performance analysis of cloudlet in Mobile Cloud Computing," *Performance Evaluation*, vol. 107, pp. 34–53, 2017.

[11] F. Mora-Gimeno, H. Mora-Mora, D. Marcos-Jorquera, and B. Volckaert, "A secure multi-tier mobile edge computing model for data processing offloading based on degree of trust," *Sensors*, vol. 18, no. 10, Article ID 3211, 2018.

[12] E. Ahmed and M. H. Rehmani, "Mobile Edge Computing: Opportunities, solutions, and challenges," *Future Generation Computer Systems*, vol. 70, pp. 59–63, 2017.

[13] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[14] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[15] L. Gupta, R. Jain, and H. A. Chan, Mobile edge computing — an important ingredient of 5g networks. IEEE Software Defined Networks, Newsletter. 2018, Available online at: http://sdn.ieee.org/newsletter/march-2016/mobile-edge-computing-an-important-ingredient-of-5g-networks.

[16] Y. Zhang, K. Wang, Y. Zhou, and Q. He, "Enhanced adaptive cloudlet placement approach for mobile application on spark," *Security and Communication Networks*, vol. 2018, Article ID 1937670, pp. 1–12, 2018.

[17] J. Pereira, L. Ricardo, M. Luís, C. Senna, and S. Sargento, "Assessing the reliability of fog computing for smart mobility applications in VANETs," *Future Generation Computer Systems*, vol. 94, pp. 317–332, 2019.

[18] O. Salman, I. Elhajj, A. Chehab, and A. Kayssi, "IoT survey: An SDN and fog computing perspective," *Computer Networks*, vol. 143, pp. 221–246, 2018.

[19] D. Rosário, M. Schimuneck, J. Camargo et al., "Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support," *Sensors*, vol. 18, no. 2, 2018.

[20] H. Mora, M. Signes-Pont, D. Gil, and M. Johnsson, "Collaborative working architecture for IoT-based applications," *Sensors*, vol. 18, no. 6, Article ID 1676, 2018.

[21] H. Mora, D. Gil, R. M. Terol, J. Azorín, and J. Szymanski, "An IoT-based computational framework for healthcare monitoring in mobile environments," *Sensors*, vol. 17, no. 10, Article ID 2302, 2017.

[22] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia IoT systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, 2018.

[23] X. Fei, N. Shah, N. Verba et al., "CPS data streams analytics based on machine learning for Cloud and Fog Computing: A survey," *Future Generation Computer Systems*, vol. 90, pp. 435–450, 2019.

[24] H. El-Sayed, S. Sankar, M. Prasad et al., "Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment," *IEEE Access*, vol. 6, pp. 1706–1717, 2017.

[25] J. I. Benedetto, L. A. González, P. Sanabria, A. Neyem, and J. Navón, "Towards a practical framework for code offloading in the Internet of Things," *Future Generation Computer Systems*, vol. 92, pp. 424–437, 2019.

[26] D. Kumar, G. Baranwal, Z. Raza, and D. P. Vidyarthi, "A survey on spot pricing in cloud computing," *Journal of Network and Systems Management*, vol. 26, no. 4, pp. 809–856, 2018.

[27] T. Wang, X. Wei, C. Tang, and J. Fan, "Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints," *Peer-to-Peer Networking and Applications*, vol. 11, no. 4, pp. 793–807, 2018.

[28] L. Liu, Q. Fan, and R. Buyya, "A deadline-constrained multi-objective task scheduling algorithm in mobile cloud environments," *IEEE Access*, vol. 6, pp. 52982–52996, 2018.

[29] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *IEEE International Symposium on Information Theory (ISIT)*, 2016.

[30] J. Zheng, Y. Cai, Y. Wu, and X. S. Shen, "Dynamic Computation Offloading for Mobile Cloud Computing: A Stochastic Game-Theoretic Approach," *EEE Transactions on Mobile Computing*, 2018.

[31] Y. Li, M. Chen, W. Dai, and M. Qiu, "Energy optimization with dynamic task scheduling mobile cloud computing," *IEEE Systems Journal*, vol. 11, no. 1, pp. 96–105, 2017.

[32] D. Rahbari, M. Nickray, and G. Heydari, "A Two-Stage Technique for Quick and Low Power Offloading in IoT," in *Proceedings of the international conference on smart cities and internet of things*, 2018.

[33] P. P. Hung and E.-N. Huh, "An adaptive procedure for task scheduling optimization in mobile cloud computing," *Mathematical Problems in Engineering*, vol. 2015, Article ID 969027, p. 13, 2015.

[34] A. Farahzadi, P. Shams, J. Rezazadeh, and R. Farahbakhsh, "Middleware technologies for cloud of things: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 176–188, 2018.

[35] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

[36] H. Flores and S. N. Srirama, "Mobile cloud middleware," *The Journal of Systems and Software*, vol. 92, no. 1, pp. 82–94, 2014.

[37] R. Schneider, D. Goswami, A. Masrur, M. Becker, and S. Chakraborty, "Multi-layered scheduling of mixed-criticality cyber-physical systems," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 1215–1230, 2013.

[38] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones, Mobile Computing," in *Int. Conf. on Mobile Computing, Applications, and Services (MobiCASE 2010)*, pp. 59–79, Springer, 2010.

[39] L. Jiang, L. D. Xu, H. Cai et al., "An IoT oriented data storage framework in cloud computing platform," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1443–1451, 2014.

[40] D. Wu, S. Liu, L. Zhang et al., "A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing," *Journal of Manufacturing Systems*, vol. 43, pp. 25–34, 2017.

[41] I. Farris, A. Orsino, L. Militano, A. Iera, and G. Araniti, "Federated IoT services leveraging 5G technologies at the edge," *Ad Hoc Networks*, vol. 68, pp. 58–69, 2018.

[42] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K.-K. R. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017.

[43] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: from concept to practice and beyond," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 80–88, 2015.

[44] D. Van Le and C.-K. Tham, "Quality of service aware computation offloading in an ad-hoc mobile cloud," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8890–8904, 2018.

[45] M. Kaya, A. Koçyiğit, and P. E. Eren, "An adaptive mobile cloud computing framework using a call graph based model," *Journal of Network and Computer Applications*, vol. 65, pp. 12–35, 2016.

[46] H. Mora, J. F. Colom, D. Gil, and A. Jimeno-Morenilla, "Distributed computational model for shared processing on Cyber-Physical System environments," *Computer Communications*, vol. 111, pp. 68–83, 2017.

[47] H. S. Narman, M. S. Hossain, M. Atiquzzaman, and H. Shen, "Scheduling internet of things applications in cloud computing," *Annals of Telecommunications*, vol. 72, no. 1-2, pp. 79–93, 2017.

[48] A. Malik and H. Om, "Cloud computing and internet of things integration: Architecture, applications, issues, and challenges," in *Sustainable Cloud and Energy Services*, pp. 1–24, 2017.

[49] X. Chen, S. Chen, X. Zeng, X. Zheng, Y. Zhang, and C. Rong, "Framework for context-aware computation offloading in mobile cloud computing," *Journal of Cloud Computing*, vol. 6, no. 1, 2017.

[50] N. Mohan and J. Kangasharju, "Edge-Fog cloud: A distributed cloud for Internet of Things computations," in *Cloudification of the Internet of Things (CIoT)*, Paris, France, 2016.

[51] M. A. Khan, "A survey of computation offloading strategies for performance improvement of applications running on mobile devices," *Journal of Network and Computer Applications*, vol. 56, pp. 28–40, 2015.

[52] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, "SmartCityWare: A service-oriented middleware for cloud and fog enabled smart city services," *IEEE Access*, vol. 5, pp. 17576–17588, 2017.

[53] B. Cao, S. Xia, Y. Li, and B. Li, "An incentive-based workload assignment with power allocation in ad hoc cloud," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, Paris, France, 2017.

[54] Z. Lv, T. Yin, X. Zhang, H. Song, and G. Chen, "Virtual reality smart city based on WebVRGIS," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1015–1024, 2016.

[55] T. Fernández-Caramés, P. Fraga-Lamas, M. Suárez-Albela, and M. Vilar-Montesinos, "A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard," *Sensors*, vol. 18, no. 6, p. 1798, 2018.

[56] H. Li, H. Wang, A. Xiong, J. Lai, and W. Tian, "Comparative analysis of energy-efficient scheduling algorithms for big data applications," *IEEE Access*, vol. 6, pp. 40073–40084, 2018.

[57] L. Zou, A. Javed, and G. Muntean, "Smart mobile device power consumption measurement for video streaming in wireless environments: WiFi vs. LTE," in *Proceedings of the 2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–6, Cagliari, Italy, June 2017.

[58] E. Filiopoulou, P. Mitropoulou, C. Michalakelis, and M. Nikolaidou, "The rise of cloud brokerage: Business model, profit making and cost savings," in *International Conference on the Economics of Grids, Clouds, Systems, and Services*, Lecture Notes in Computer Science, pp. 19–32, 2017.

[59] E. Casalicchio, V. Cardellini, G. Interino, and M. Palmirani, "Research challenges in legal-rule and QoS-aware cloud service

brokerage," *Future Generation Computer Systems*, vol. 78, pp. 211–223, 2018.

[60] Y. Jararweh, L. Tawalbeh, F. Ababneh, A. Khreishah, and F. Dosari, "Scalable cloudlet-based mobile computing model," *Procedia Computer Science*, vol. 34, pp. 434–441, 2014.